The Definition of Transitive Closure with OCL – Limitations and Applications –

Thomas Baar

Swiss Federal Institute of Technology Lausanne (EPFL)
Software Engineering Laboratory
CH-1015 Lausanne EPFL, Switzerland
Email: thomas.baar@epfl.ch

Abstract. The Object Constraint Language (OCL) is based on first-order logic and set theory. As the most well-known application, OCL is used to formulate well-formedness rules in the UML metamodel. Here, the transitive closure of a relationship is defined in terms of an OCL invariant, which seems to contradict classical results on the expressive power of first-order logic.
In this paper, we give sufficient justification for the correctness of the definition of transitive closure. Our investigation reinforces some decisions made in the semantics of UML and OCL. Currently, there is a lively debate on the same issues in the semantics of the upcoming UML 2.0.

1 Introduction

The Object Constraint Language (OCL) is a textual language to annotate UML diagrams (see [5] for a detailed introduction to syntax and application of OCL). For the purpose of this paper we restrict ourselves to invariant expressions in OCL.

The semantics of a UML class diagram is defined formally by the set of admissible object diagrams, where classes, attributes, associations are realized by sets of objects, slots, links [3, 4]. Sometimes, an object diagram is called a *state* or *state configuration*.
The semantics of OCL constraints is given in [4] by a formally defined evaluation function which maps, in a given state, any OCL constraint to one of the logical constants `true`, `false`, `undefined`. [1] In admissible states all invariants of the corresponding class diagram must be evaluated to `true`. Certain restrictions on OCL's syntax ensure that the evaluation of each OCL expression must terminate.

Since every object diagram is admissible unless an invariant is evaluated to `false` or `undefined`, OCL has a *loose semantics*. A loose semantics is natural in the sense that the classical first-order logic (FOL) is semantically defined in the same way. There are several possibilities for translations of OCL constraints into first-order formulas (see Section 2). Classes in UML class diagrams are typically

[1] The semantics [4] by Richters was adopted by the official language specification, cmp. [3, Chap. 6].

mapped to types, OCL variables, such as `self`, are mapped to variables with restricted quantification, attributes are mapped to function symbols, etc.
At a first glance, the translation into FOL allows to lift up classical results from FOL to OCL. For example, the logical entailment relation on OCL constraints seems to be semi-decidable, since we can translate invariants into FOL formulas (GÖDEL's Completeness Theorem, 1930). This argumentation ignores a side-condition of UML semantics stipulating object diagrams to contain only finitely many objects. Thus, the translation of invariants into FOL formulas is only correct if the semantics of FOL formulas is restricted to finite models.[2] This makes deduction on OCL invariants more difficult since GÖDEL's Completeness Theorem is not valid on finite models. Moreover, a sound and complete calculus does not exist (TRACHTENBROT, 1950).

Other results from classical logic can certainly be lifted from OCL to FOL, e.g. the transitive closure of a binary relation cannot be formalized using FOL regardless of a restriction to finite models.[3]

The metamodel of UML as part of the official language description [3] describes the abstract syntax of UML diagrams in terms of a class diagram and OCL well-formedness rules. Class diagrams are basically defined as graphs, where classes are represented as nodes and associations/generalizations are represented as edges. In order to describe the type system of UML, a class 'has to know' all its superclasses, i.e. the transitive closure of its direct superclasses. In the metamodel, the set of all superclasses is defined using an OCL invariant, which at a first glance contradicts the classical result on the undefinability of transitive closure by FOL formulas. Even so, it is often discussed in the OCL community to substitute, within the metamodel, that definition of transitive closure by informal text, because for 'theoretical reasons' or so we are told.

In this paper, we prove the invariant defining the set of all supertypes to be fully correct. We try to clarify common misunderstandings of classical results by presenting countermodels for apparent FOL definitions of transitive closure. We also discuss the reason why the countermodels do not apply to the definition of all supertypes in the metamodel.

The paper is organized as follows: Section 2 presents the relevant part of the UML metamodel. It also gives a translation into FOL formulas, which provides a more abstract view on the problem. Section 3 enlists countermodels for 'FOL definitions of transitive closure' together with theorems to avoid countermodels. Finally, we draw a conclusion in Section 4.

[2] Throughout the paper, we use the term *model* with the meaning it has in classical logic. A *UML model* is in our terminology just a concrete class diagram.

[3] An alternative formulation, of the same result, is the inability of FOL formulas to axiomatize connected, ordered graphs. For finite models, this can be proven using EHRENFEUCHT-FRAÏSSÉ games (see [2] for a detailed account). If infinite models are allowed, this is a consequence of the Compactness Theorem.

2 Transitive Closure in the UML Metamodel

The UML metamodel has proved to be a very intuitive, concise and precise way to describe the syntax of UML diagrams. Thus, the metamodel has the same purpose as an EBNF grammar for formal languages. Also the vocabulary we use in EBNF grammars and in the metamodel are similar syntactical terms, e.g., a grammar for JAVA contains nonterminals like *Assignment*, *Statement*, *Block*, where in the metamodel similar syntactical terms, such as *Class*, *Classifier*, *Association* are realized as metaclasses.

Figure 1 presents a very small part of the metamodel (cmp. [3, p. 2-14]) and can be read as follows. A *class* in a UML diagram is a special kind of a *classifier* what in turn is a special kind of a *generalizable element*. Each generalizable element can be linked to unspecified many generalizations which in turn are linked to exactly two generalizable elements called `child` and `parent`.[4]

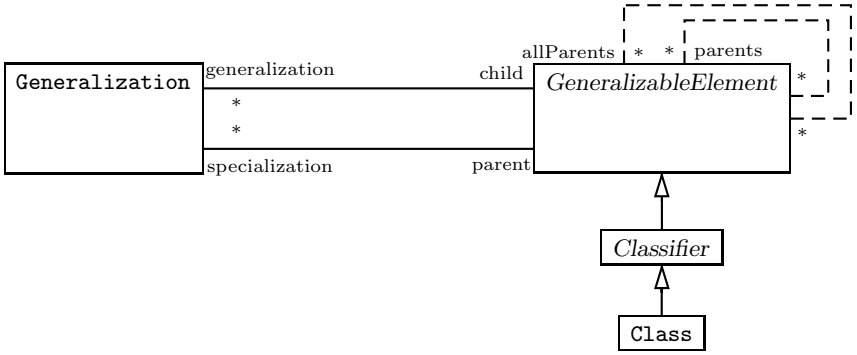


Fig. 1. Part of the UML metamodel, package Core

The dashed associations in Figure 1 are defined in the metamodel by auxiliary definitions (cmp. [3, p. 2-61]):

```
context GeneralizableElement
    def: parents: Set(GeneralizableElement) =
           self.generalization->collect(gen| gen.parent)

    def: allParents: Set(GeneralizableElement) = self.parents->
           union(self.parents->collect(ge| ge.allParents))
```

Informally speaking, `self.parents` denotes the set of all direct supertypes and `self.allParents` denotes the transitive closure of direct supertypes.

It is easy to read the last invariant accordingly to the formal semantics of OCL given in [4] as a mathematical expression (a full translation of OCL and UML into FOL can be found in [1]). Let x, y, z be variables for generalizable

[4] In class diagrams, generalizations are symbolized by generalization arrows from the subclass (`child`) to the superclass (`parent`).

elements, $Par(x)$ the translation of `x.parents`, and $APar(x)$ the translation of `x.allParents`. Then, the OCL definition of `allParents` can be written as (we substitute the OCL variable `self` by x, which is a more common variable name in mathematics):

$$APar(x) = Par(x) \cup \{y \mid \exists z\ z \in Par(x) \wedge y \in APar(z)\}$$

This can be further transformed into pure FOL where Par, $APar$ are substituted by two new relation symbols r and r^* with $r(x,y) \leftrightarrow y \in Par(x)$ and $r^*(x,y) \leftrightarrow y \in APar(x)$:

$$r^*(x,y) \leftrightarrow r(x,y) \vee (\exists z\ r(x,z) \wedge r^*(z,y)) \tag{DEF}$$

3 Definition of Transitive Closure and Countermodels

Formula (DEF) is a FOL version of the attempt to define the transitive closure with OCL. Now, we can analyse (DEF) and can construct countermodels where the interpretation of r^* is not the transitive closure of the interpretation of r. Although our argumentation is based on the FOL formula (DEF) it clearly applies to the original OCL formalization as well. Every countermodel of (DEF) can easily be transformed into a non-intended object diagram. Countermodels are presented using the following notation.

Notation 1
The formula (DEF) is always interpreted by the structure (U, R, R^*) where U is the universe and the relations R, R^* are interpretations of the relation symbols r, r^*, respectively. We are only interested in models of (DEF). Thus, we always have $R \subseteq R^*$. Elements of U are denoted by a, b, c, possibly decorated with subscripts. A pair (a, b) being an element of a relation S is often denoted as aSb instead of the mathematical notation $(a, b) \in S$.

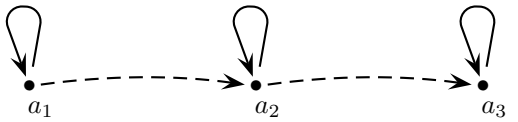
There is also a graphical notation for structure (U, R, R^*). The elements of U are displayed as labelled nodes, the elements of R by solid arcs. The elements of R^* not belonging to R are displayed by dashed arcs.

The transitive closure of a relation S is denoted as $TC(S)$ and defined as usual. Please note, that $TC(S)$ is transitive for any S.

$TC(S) = \{(a,b) \mid \exists a_1 \ldots a_n\ a = a_1 \wedge b = a_n \wedge a_i S a_{i+1}\ for\ i = 1, \ldots, n-1\}$

The next example presents a model of (DEF) but $R^* \neq TC(R)$. Thus, the formula (DEF) is not a FOL formalization of transitive closure.

Example 1 (Reflexive Countermodel for (DEF))



Obviously, the relation R^* cannot be the same as $TC(R)$ since R^* is not transitive (the pair (a_1, a_3) is missing).

In the countermodel of Example 1, the relation R^* does not coincide with $TC(R)$ but R^* comprises $TC(R)$. It might arise the question whether this is true in every case. The next Theorem 1 gives an answer and characterizes all models of (DEF) in this respect.

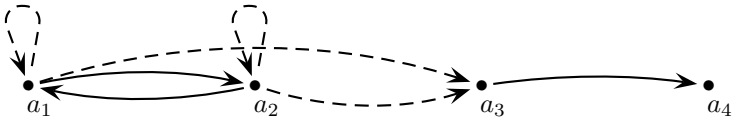
THEOREM 1 (R^* COMPRISES TRANSITIVE CLOSURE)
Let (U, R, R^*) be a model of (DEF). Then,

$$TC(R) \subseteq R^*$$

So far, we have investigated the model of (DEF) without making any assumptions on the interpretation of r. Surely, the formula (DEF) can be fixed in a way that would prevent the countermodel given in Example 1. One attempt could be:

$$r^*(x,y) \leftrightarrow r(x,y) \vee (\exists z\ x \neq z \wedge r(x,z) \wedge r^*(z,y)) \qquad \text{(DEF')}$$

However, from the classical results cited in the introduction, we can conclude the existence of countermodels for any such 'improved formulas'. In the case of (DEF'), cycles of length two are possible in r, which allows as a countermodel for (DEF'):



Apparently, this is a hopeless situation, but we should not give up too early. The situation becomes much more comfortable, if we can restrict the interpretation of r appropriately. Suppose, the following axiom is given as an additional information on r:

$$\exists x_1 x_2\ \forall x\ (x = x_1 \vee x = x_2) \wedge \forall yz\ (r(y,z) \leftrightarrow y = x_1 \wedge z = x_2) \qquad \text{(FIXR)}$$

If (FIXR) is valid, the formula (DEF) is a correct definition of transitive closure because in all models of (FIXR, DEF) we have obviously $R = TC(R) = R^*$. Doubtless, the axiom (FIXR) is a very strong restriction on R, but it illustrates the main principle of winning expressiveness by sacrificing universality.

The next Theorem 2 exploits the very same idea and makes two sufficient assumptions explicit. If the models are restricted to be finite and the transitive closure of R does not have any cycles, then (DEF) is a correct definition of the transitive closure. Note, that this assumption on R is a real restriction. For instance, any relation R containing a reflexive pair (a, a) would be not allowed as an interpretation of r.

THEOREM 2 (SOUNDNESS FOR FINITE MODELS, NON-CYCLIC RELATIONS)
Let (U, R, R^*) be a finite model of both (DEF) and axiom (NONCYC):

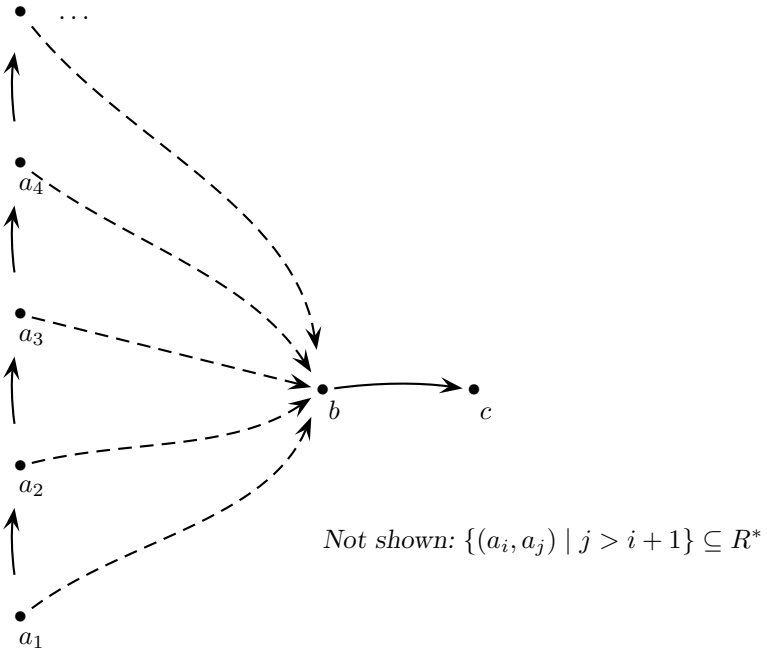
$$\neg r^*(x,x) \qquad \text{(NONCYC)}$$

Then,

$$TC(R) = R^*$$

Please note, that the finiteness of the model is an essential assumption. If infinite models are allowed, there is a countermodel for (DEF) and (NONCYC) as the next example shows.

Example 2 (counter model on infinite universe)



In the counter model, an infinite sequence of elements $a_1, a_2, a_3, \ldots$ is assumed where $a_i R a_{i+1}$ for all $i = 1, 2, \ldots$
The relation R^* is not transitive (all pairs (a_i, c) are missing) but have all properties expressed in axioms (DEF) and (NONCYC).

The Theorem 2 justifies to call (DEF) a correct axiomatization for transitive closure presuming the two preconditions to be valid. Fortunately, the two preconditions are satisfied for the definition of `allParents` in the UML metamodel.

Corollar 1 (Correct Definition of `allParents`)
For any object of class `GeneralizableElement` the evaluation of the OCL expression `self.allParents` is always the transitive closure of `self.parents` in any admissible object diagrams for the UML metamodel.

Proof:

We only have to prove that the preconditions of Theorem 2 are satisfied. Clearly, the object diagrams are always finite since the semantics of UML restricts the interpretation of classes to finite sets of objects.
The justification of (NONCYC) has not been mentioned so far, but (NONCYC) is intended for the supertype relationship. Thus, there is a corresponding OCL invariant in the UML metamodel (cmp. [3, p. 2-61]):

```
context GeneralizableElement
    not self.allParents->includes(self)
```

4 Conclusion

In this paper we have elaborated a problem which has already caused a lot of confusion and discussions within the OCL community.

Starting with the definition of `allParents` in the UML metamodel we investigated a given formalization of the transitive closure. By translation of invariants into first-order logic, the problem was reduced to the axiomatization of transitive closure in first-order logic.

The classical results were cited. Since software engineers often tend to misunderstand the classical results, we have presented in detail countermodels of the apparent formalization (DEF). We have proven essential properties of all models of (DEF) and pointed out additional preconditions which are sufficient to call (DEF) a correct formalization of the transitive closure. Fortunately, due to the semantics of UML and the OCL invariants given as well-formedness rules in the metamodel, these preconditions are satisfied for our motivating example, the definition of `allParents` in the metamodel.

Our investigation allows also a judgement on the semantics of UML and OCL. As said in the introduction, the restriction to interpret the classes of class diagrams only by finite sets of objects makes things more complicated from the viewpoint of deduction. On the other hand, this decision is an essential precondition to prove the `allParents` definition being correct.

Finally, our investigation has presented a meaningful recursive definition in OCL (note, the definition of `allParents` is recursive). This is worth to mention because OCL has currently a simple loose semantics but there is a discussion to substitute the loose semantics by a more complicated fixpoint semantics. At least for the definition of `allParents`, this is absolutely unnecessary.

Acknowledgement

I would like to thank Peter H. Schmitt for critical comments on the topic of this paper and Shane Sendall for fruitful discussions.

References

1. B. Beckert, U. Keller, and P. H. Schmitt. Translating the Object Constraint Language into first-order predicate logic. In *Proceedings, VERIFY, Workshop at Federated Logic Conferences (FLoC), Copenhagen, Denmark*, 2002.
2. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer, 1995.
3. OMG. OMG Unified Modeling Language Specification. Technical Report OMG-UML Version 1.4, Object Mangagement Group, September 2001.
4. M. Richters. *A precise approach to validating UML models and OCL constraints.* PhD thesis, Bremer Institut für Sichere Systeme, Universität Bremen, Logos-Verlag, Berlin, 2001.
5. J. Warmer and A. Kleppe. *The Object Constraint Language: Precise Modeling with UML.* Addison-Wesley, 1998.

A Proofs of Theorems

THEOREM 1 (R^* COMPRISES TRANSITIVE CLOSURE)
Let (U, R, R^*) be a model of (DEF). Then,

$$TC(R) \subseteq R^*$$

Proof:

Indirectly: Let (a, b) be a pair with $(a, b) \in TC(R)$ and $\neg aR^*b$.
By definition of TC there is n, and there are $a = a_0, a_1, \ldots, a_n = b$ such that a_iRa_{i+1}. By applying (DEF) repeatly we get aR^*a_i for all $i = 0, \ldots, n$ contradicting the assumption.

THEOREM 2 (SOUNDNESS FOR FINITE MODELS, NON-CYCLIC RELATIONS)
Let (U, R, R^*) be a finite model of both (DEF) and axiom (NONCYC):

$$\neg r^*(x, x) \tag{NONCYC}$$

Then, $$TC(R) = R^*$$

Proof:

By Theorem 1 we know $TC(R) \subseteq R^*$. It remains to show $R^* \subseteq TC(R)$. Indirectly. Assume, there is a pair (a_1, b) with

$$a_1R^*b \tag{1}$$

and

$$(a_1, b) \notin TC(R) \tag{2}$$

By (2) we know $(a_1, b) \notin R$ and can conclude from (1), (DEF) that there is a_2 with

$$a_1Ra_2 \wedge a_2R^*b \tag{3}$$

and furthermore by (2)

$$(a_2, b) \notin TC(R) \tag{4}$$

These arguments can be iterated and there are $a_3, a_4, a_5, \ldots$ with

$$a_iRa_{i+1} \wedge a_{i+1}R^*b \wedge (a_{i+1}, b) \notin TC(R) \; for \; i = 2, 3, 4, \ldots \tag{5}$$

Since we have assumed U to be finite, there are a_j, a_k with $j < k \leq \|U\| + 1$ and $a_j = a_k$. However, by (5) and (DEF) we can conclude $a_jR^*a_j$ contradicting axiom (NONCYC).